WEB OF TRUST 2025

TOOLS, TECHNIQUES, AND PROCEDURES FOR

BUILDING TRUST AND SECURING THE SOFTWARE SUPPLY CHAIN

BY ME@MATTBORJA.DEV

INTRODUCTION

ORCID
stands for
Open Researcher and Contributor ID

Industry

Role

Recent Work

"SIG3"

NIST
SP 800-63A

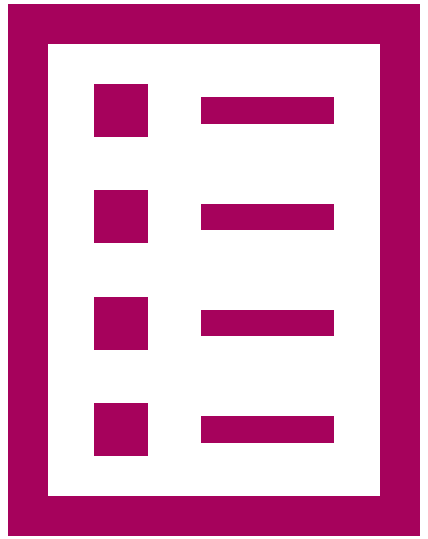
Topic

Matt Borja <me@mattborja.dev>

99BB 608E 3038 0C45 1952 D6BB
A1C7 E813 F160 A407

OBJECTIVES

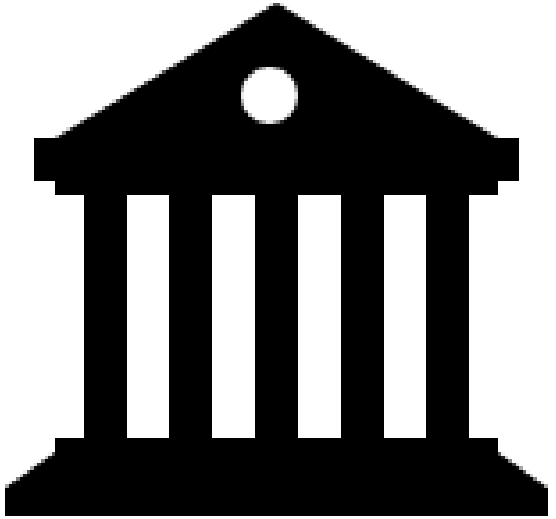
- Define conditions for trust
- Demonstrate signature verification
- Create a new identity for digital signing using GnuPG
- Share your public key
- Sign others' public key
- Understand the responsibility of auditing signing keys
- Identify numerous industry use cases for GPG and web of trust



LEGAL (COMPLIANCE)

By continuing, you acknowledge all of the following:

- Any software and technical information published or referenced herein may be subject to the U.S. Export Administration Regulations (the “EAR”) and other U.S. and foreign laws and may not be exported, re-exported or transferred
  - to a prohibited destination country under the EAR or U.S. sanctions regulations (currently Cuba, Iran, North Korea, Syria, and the Crimea Region of Ukraine, subject to change as posted by the United States government);
  - to any prohibited destination or to any end user prohibited from participating in U.S. export transactions by any federal agency of the U.S. government; or
  - for use in connection with the design, development or production of nuclear, chemical or biological weapons, or rocket systems, space launch vehicles, or sounding rockets, or unmanned air vehicle systems.
- These terms are subject to change with or without notice.



WHAT IS WEB OF TRUST?

“As time goes on, you will accumulate keys from other people that you may want to designate as trusted introducers. Everyone else will each choose their own trusted introducers. And everyone will gradually accumulate and distribute with their key a collection of certifying signatures from other people, with the expectation that anyone receiving it will trust at least one or two of the signatures. This will cause the emergence of a decentralized fault-tolerant **web of confidence** for all public keys.”

— PGP creator, Phil Zimmermann (p. 31, The Official PGP User’s Guide, 1995)

SCENARIO: KEYS

(WHAT IS WEB OF TRUST?)

Given:

- Person B — Someone you **know** and **trust** (integrity)
- Person X — A complete stranger

Explain:

- Your default confidence in X.
- Assume Person B discloses their familiarity with X to you. How does this change your disposition towards X?

Consider the impact of Person B.

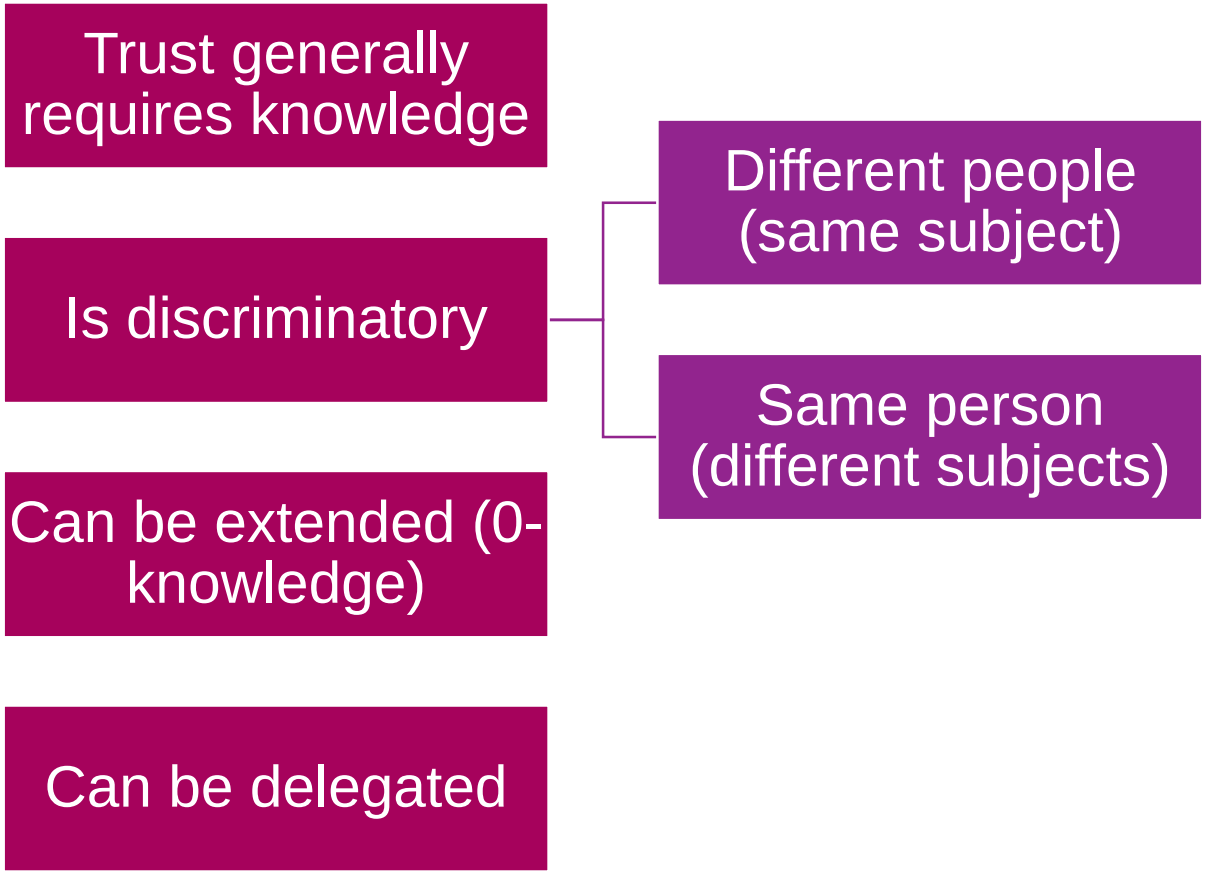
Transitive Property

CHARACTERISTICS

(WHAT IS WEB OF TRUST?)

- Person B as someone you know
- Person B as someone you would trust
- Complete stranger X
- Confidence in X
- Person B's confidence in X

- When *wouldn't* you trust Person B?



WHAT IS OPENPGP/GPG?

"OpenPGP software (message format) uses a combination of strong public-key and symmetric cryptography to provide security services for electronic communications and data storage. These services include **confidentiality**, **key management**, **authentication**, and **digital signatures**."
— RFC4880

"GnuPG (or GPG) is **a complete and free implementation** of the OpenPGP standard as defined by RFC4880 (also known as PGP). GnuPG allows you to encrypt and sign your data and communications; it features a versatile key management system, along with access modules for all kinds of public key directories."
— https://gnupg.org

WHAT IS OPENPGP/GPG?

Capability	Duration	Description
[C]ertify	Long-term	**Primary key.** This is your [offline] primary key which your identity is based on. Used to sign (certify) other keys and manage subkeys.
[S]ign*	1 year	Subkey. Sign messages, files, etc. (integrity)
[E]ncrypt*	1 year	Subkey. Encrypt messages, files, etc. (confid.)
[A]uthenticate*	1 year	Subkey. Authentication (e.g., SSH).
[SEA]	1 year	Combined: Sign, Encrypt, Authenticate

* **Important Disclaimer:**
GnuPG does NOT support Perfect Forward Secrecy (PFS)!

VERIFYING SIGNATURES

A signature is a verifiable attestation that gives a recipient confidence in the origin of a message.

Requirements

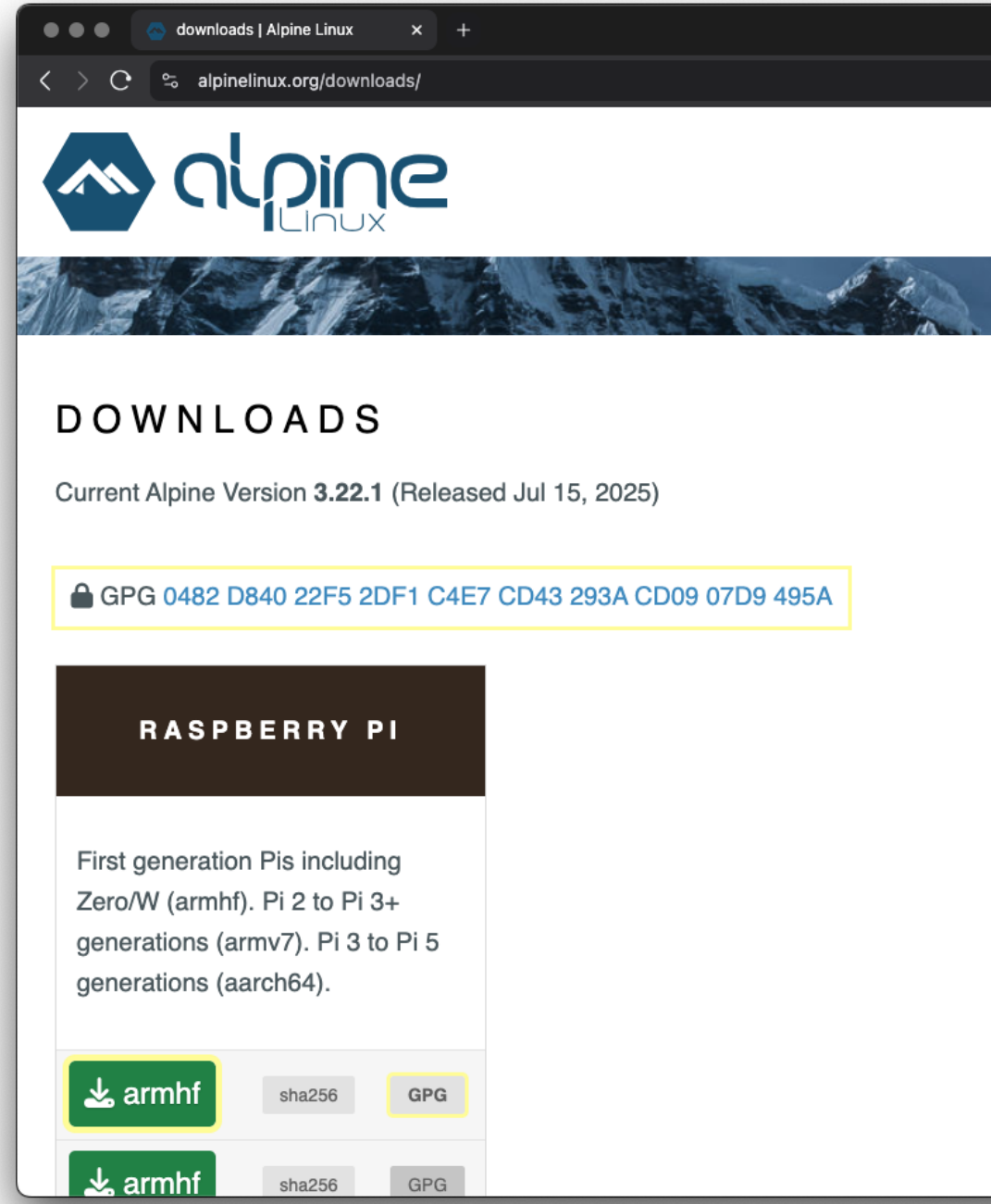
- Artifact (message)
- Signature
- Public signing key
- gpg (command line)

Procedure

- Validate the artifact's signature
- Verify the signature's origin (UID & fingerprint)

EXAMPLE: ALPINE LINUX
(VERIFYING SIGNATURES)

- Visit **https://www.alpinelinux.org**
- Click on Downloads
- Note GPG fingerprint (copy)
- Click the GPG fingerprint (download)
- Download an *artifact* to verify (e.g., *armhf*)
- Download its corresponding GPG signature
- Import the public key (**`gpg --import`**)
- Verify the signature (**`gpg --verify`**)



EXAMPLE: ALPINE LINUX
(VERIFYING SIGNATURES)

Download and verify the signature against its artifact (without public key):

```
$ gpg --verify alpine-rpi-3.22.0-armhf.img.gz.asc alpine-rpi-3.22.0-armhf.img.gz
                              signature                        artifact

gpg: Signature made Sat Jul 15 04:50:27 2025 MST
gpg:                using RSA key 0482D84022F52DF1C4E7CD43293ACD0907D9495A
⚠ gpg: Can't check signature: No public key
```

💡 Where does GPG look for public keys?

Linux/Mac: **~/.gnupg/pubring.kbx** Windows: ~/AppData/Roaming/gnupg/trustdb.gpg

EXAMPLE: ALPINE LINUX
(VERIFYING SIGNATURES)

Download and import the hyperlinked public key ("ncopa.asc"):

```
$ gpg --import ./ncopa.asc

gpg: key 293ACD0907D9495A: public key "Natanael Copa <ncopa@alpinelinux.org>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

EXAMPLE: ALPINE LINUX
(VERIFYING SIGNATURES)

Verify the signature against its artifact:

```
$ gpg --verify alpine-rpi-3.22.0-armhf.img.gz.asc alpine-rpi-3.22.0-armhf.img.gz
                              signature                      artifact

gpg: Signature made Sat Jul 15 04:50:27 2025 MST
gpg:                using RSA key 0482D84022F52DF1C4E7CD43293ACD0907D9495A
gpg: Good signature from "Natanael Copa <ncopa@alpinelinux.org>" [unknown]
⚠ gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 0482 D840 22F5 2DF1 C4E7  CD43 293A CD09 07D9 495A
```

EXAMPLE: ALPINE LINUX
(VERIFYING SIGNATURES)

Inspect key certifications (UID signatures):

```
$ gpg --list-signatures "0482 D840 22F5 2DF1 C4E7 CD43 293A CD09 07D9 495A"
                                                        64-bit Key ID

pub   rsa4096 2014-12-10 [SC]
      0482D84022F52DF1C4E7CD43293ACD0907D9495A
uid           [ unknown] Natanael Copa <ncopa@alpinelinux.org>     IAL 0 (NIST SP 800-63A)
sig 3        293ACD0907D9495A 2014-12-10  [self-signature]
sub   rsa4096 2014-12-10 [E]
sig          293ACD0907D9495A 2014-12-10  [self-signature]
```

EXAMPLE:
ALPINE LINUX
(VERIFYING SIGNATURES)

(!) TODO — Research must be done to prove the following statement:

Key ID

293A CD09 07D9 495A

(0482 D840 22F5 2DF1 C4E7 CD43 293A CD09 07D9 495A)

belongs to UID

Natanael Copa <ncopa@alpinelinux.org>

? What are some ways you can verify this statement?

(NIST SP 8003A)

Recent Work

"SIG3"

EXAMPLE: SHIBBOLETH IDP (SSO)

(THREAT MODEL)

Identity Provider

https://shibboleth.net/downloads

```
$ gpg --verify shib-idp-5.1.4-x64.msi.asc shib-idp-5.1.4-x64.msi

gpg: Signature made Thu Mar 27 06:23:34 2025 MST
gpg:                using RSA key 4AF4D83EEDDF43DA3C06CB3101483F262A4B3FF0
gpg: Good signature from "Rod Widdowson <rdw@steadingsoftware.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 4AF4 D83E EDDF 43DA 3C06  CB31 0148 3F26 2A4B 3FF0
```

[C]

⚠ ACADEMIC USE ONLY!

NEW IDENTITY
(DIGITAL SIGNING)

Create a new PGP key set with your UID:

```
$ gpg --expert --full-generate-key
```

```
Please select what kind of key you want:
   (1) RSA and RSA
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
   (7) DSA (set your own capabilities)
   (8) RSA (set your own capabilities)
   (9) ECC (sign and encrypt) *default*
  (10) ECC (sign only)
  (11) ECC (set your own capabilities)
  (13) Existing key
  (14) Existing key from card
Your selection? 11
```

Pro Tip!

These are the same keygen steps for creating subkeys [S, E, A] later on.

[C]

ACADEMIC USE ONLY!

NEW IDENTITY
(DIGITAL SIGNING)

```
Possible actions for this ECC key: Sign Certify Authenticate
Current allowed actions: Sign Certify
```

Capability	Duration	Description
[C]ertify	Long-term	**Primary key.** This is your offline identity key which your fingerprint is based on. Used to sign (certify) other keys and generate new subkeys.
[S]ign	1 year	Subkey. Used to sign messages, files, etc.
[E]ncrypt	1 year	Subkey. Used to encrypt messages, files, etc.
[A]uthenticate	1 year	Subkey. Used for authentication (e.g., SSH).
[SEA]	1 year	Combined. Sign, Encrypt, Authenticate

[C]

ACADEMIC USE ONLY!

NEW IDENTITY
(DIGITAL SIGNING)

Specify cryptographic capabilities:

```
Possible actions for this ECC key: Sign Certify Authenticate
Current allowed actions: Sign Certify

    (S) Toggle the sign capability
    (A) Toggle the authenticate capability
    (Q) Finished

Your selection? S
```

[C]

ACADEMIC USE ONLY!

NEW IDENTITY
(DIGITAL SIGNING)

Specify cryptographic capabilities:

```
Possible actions for this ECC key: Sign Certify Authenticate
Current allowed actions: Certify

   (S) Toggle the sign capability
   (A) Toggle the authenticate capability
   (Q) Finished

Your selection? Q
```

[C]

ACADEMIC USE ONLY!

NEW IDENTITY
(DIGITAL SIGNING)

```
Please select which elliptic curve you want:
    (1) Curve 25519 *default*
    (2) Curve 448
    (3) NIST P-256
    (4) NIST P-384
    (5) NIST P-521
    (6) Brainpool P-256
    (7) Brainpool P-384
    (8) Brainpool P-512
    (9) secp256k1
Your selection? 1
```

[C]

ACADEMIC USE ONLY!

NEW IDENTITY

(DIGITAL SIGNING)

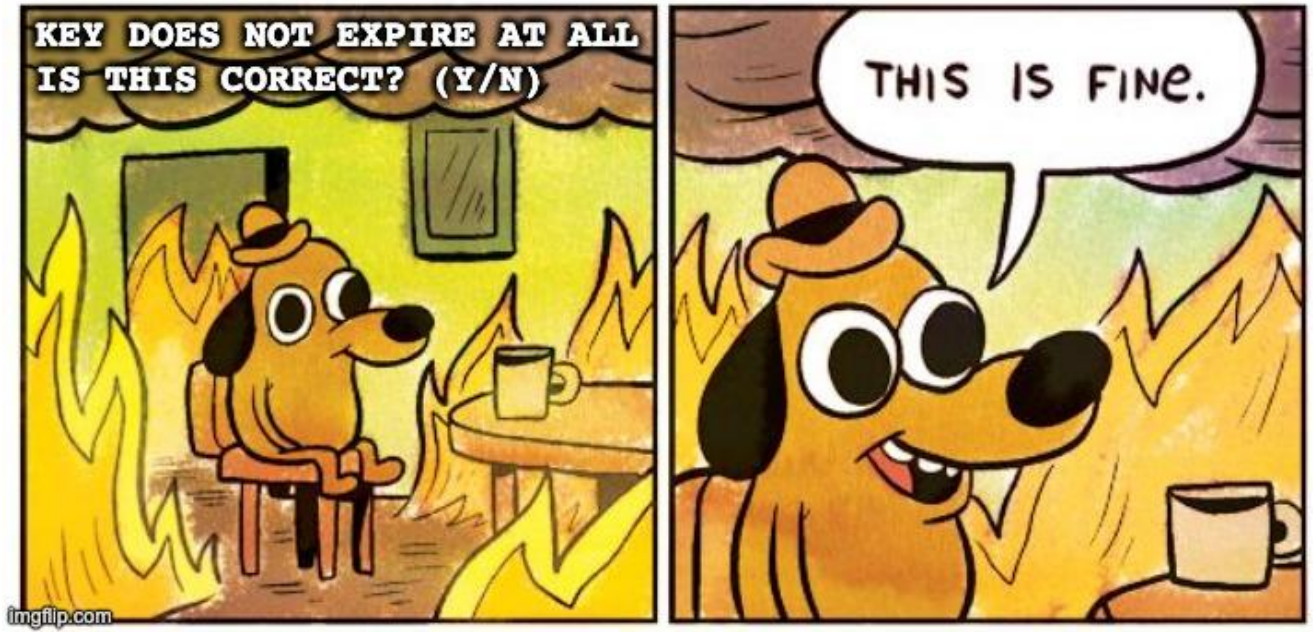
```
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years

Key is valid for? (0)
```

[C]

ACADEMIC USE ONLY!

NEW IDENTITY

(DIGITAL SIGNING)



[C]

ACADEMIC USE ONLY!

NEW IDENTITY
(DIGITAL SIGNING)

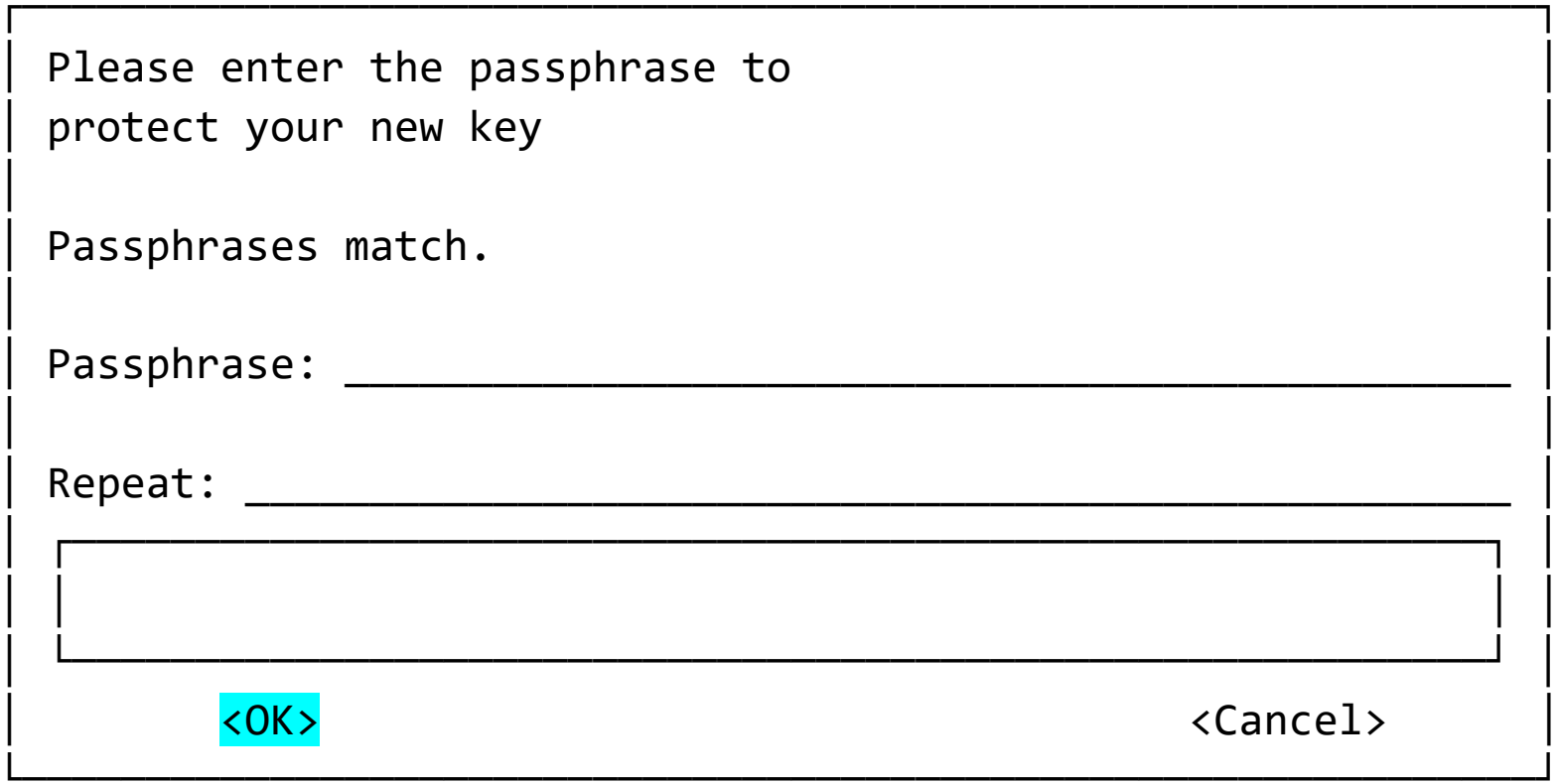
```
GnuPG needs to construct a user ID to identify your key.

Real name: Matt Borja
Email address: me@mattborja.dev
Comment: First GPG key. For academic use only!
You selected this USER-ID:
    "Matt Borja (First GPG key. For academic use only!) <me@mattborja.dev>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
```

[C]

ACADEMIC USE ONLY!

NEW IDENTITY

(DIGITAL SIGNING)



[C]

ACADEMIC USE ONLY!

NEW IDENTITY
(DIGITAL SIGNING)

```
You have not entered a passphrase - this is in general a bad idea!
Please confirm that you do not want to have any protection on your key.

<Yes, protection is not needed>                    <Enter new passphrase>
```

[C]

ACADEMIC USE ONLY!

NEW IDENTITY
(DIGITAL SIGNING)

```
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
[1]  gpg: revocation certificate stored as '~/.gnupg/openpgp-revocs.d/...E87215B3E9FA14FA.rev'
[2]  public and secret key created and signed.
[3]  pub   ed25519 2025-07-06 [C]
[4]        CC7079F2988A849BCF39AC3BE87215B3E9FA14FA
[5]  uid                      Matt Borja (First GPG key. For academic use only!) <me@mattborja.dev>
```

This is your new [C]ertify key and main identity.

[C]

ACADEMIC USE ONLY!

NEW IDENTITY
(DIGITAL SIGNING)

Share your public key:

```
$ gpg --armor --export CC7079F2988A849BCF39AC3BE87215B3E9FA14FA
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mDMEaGrCcRYJKwYBBAHaRw8BAQdAX24jS+XmDnFXbI1PbcMBFbQze50NjvqAna9X
iRj3shC0RU1hdHQgQm9yamEgKEZpcnN0IEdQRyBrZXkuIEZvciBhY2FkZW1pYyB1
c2Ugb25seSEpIDxtZUBtYXR0Ym9yamEuZGV2PoiTBBMWCgA7FiEEzHB58piKhJvP
Oaw76HIVs+n6FPoFAmhqxIUCGwEFCwkIBwICIgIGFQoJCAsCBBYCAwECHgcCF4AA
CgkQ6HIVs+n6FPqSWQEA2UgfYwFOCrVKFeXkAOpw9e2wm7btCCKwlJ3xkt0YloEB
AKqSdFFBxELW/7cLdCUV891cnDf+YLaraDgOIAEUZnsAMdGv
-----END PGP PUBLIC KEY BLOCK-----
```

Copy:
CTL + C
⌘ + C

Then,
email it!

Or...

[C]

NEW IDENTITY
(SHARING YOUR PUBLIC KEY)

- Login to GitHub.com
- Go to **Account Settings**
- Click on **SSH and GPG keys** (under Access)
- Click **New GPG key**
- Paste your new GPG key
- Provide a title
- Click **Add GPG key** to save

Add new GPG key:
https://github.com/settings/gpg/new

Result (Share URL):
https://github.com/username.gpg

Enable #two-factor-authentication!
https://github.com/settings/security

(Because MFA protected GPG keys = 🔥🔥🔥)

[C]

NEW IDENTITY

(SHARING YOUR PUBLIC KEY)

Power Users: To inspect a key before importing...

```
$ curl -s https://github.com/username.gpg | gpg --list-packets
# off=0 ctb=98 tag=6 hlen=2 plen=51
:public key packet:
version 4, algo 22, created 1728328837, expires 0
...
keyid: A1C7E813F160A407
# off=53 ctb=b4 tag=13 hlen=2 plen=63
:user ID packet: "Matt Borja (Offline long-lived identity key) <me@mattborja.dev>"
# off=118 ctb=88 tag=2 hlen=2 plen=150
:signature packet: algo 22, keyid A1C7E813F160A407
    version 4, created 1728328837, md5len 0, sigclass 0x13
    digest algo 10, begin of digest a1 4e
    hashed subpkt 33 len 21 (issuer fpr v4 99BB608E30380C451952D6BBA1C7E813F160A407)
    hashed subpkt 2 len 4 (sig created 2024-10-07)
```

[C]

WEB OF TRUST

(SIGNING PUBLIC KEYS)

i [C]ertify key required!

- Import the public key to be signed
- Edit the public key being signed
- **Review UID information**
- Confirm fingerprint ownership
- Use the **sign** subcommand
- Review confirmation prompt, entering '?' for additional guidance
- Specify a signature level (0-3)
- Enter **save** to commit and exit
- Verify with **`gpg --list-sigs`**
- Export signed public key

```
$ gpg --import 01483F262A4B3FF0.asc
...

$ gpg -u E87215B3E9FA14FA --ask-cert-level --edit-key
01483F262A4B3FF0

gpg (GnuPG) 2.4.7; Copyright (C) 2024 g10 Code GmbH
This is free software: you are free to change and
redistribute it.
There is NO WARRANTY, to the extent permitted by law.

pub  rsa4096/01483F262A4B3FF0
     created: 2019-09-18  expires: never   usage: SC
     trust: unknown           validity: unknown
sub  rsa4096/D77D4942441D628D
     created: 2019-09-18  expires: never   usage: E
[  full  ] (1). Rod Widdowson <rdw@steadingsoftware.com>

gpg> fpr
pub   rsa4096/01483F262A4B3FF0 2019-09-18 Rod Widdowson
<rdw@steadingsoftware.com>
 Primary key fingerprint: 4AF4 D83E EDDF 43DA 3C06  CB31
0148 3F26 2A4B 3FF0
```

[C]

WEB OF TRUST
(SIGNING PUBLIC KEYS)

- Import the public key to be signed
- Edit the public key being signed
- Review UID information
- Confirm fingerprint ownership
- Use the **sign** subcommand
- Review confirmation prompt, entering '?' for additional guidance
- Specify a signature level (0-3)
- Enter **save** to commit and exit
- Verify with `gpg --list-sigs`
- Export signed public key

```
...
gpg> sign

pub  rsa4096/01483F262A4B3FF0
     created: 2019-09-18   expires: never   usage: SC
     trust: full           validity: full
Primary key fingerprint: 4AF4 D83E EDDF 43DA 3C06  CB31
0148 3F26 2A4B 3FF0

     Rod Widdowson <rdw@steadingsoftware.com>

  How carefully have you verified the key you
  are about to sign actually belongs
  to the person named above?  If you don't know
  what to answer, enter "0".

     (0) I will not answer.
     (1) I have not checked at all.
     (2) I have done casual checking. (default)
     (3) I have done very careful checking.

  Your selection? (enter '?' for more information): ?
```

[C]

WEB OF TRUST
(SIGNING PUBLIC KEYS)

When you sign a user ID on a key, you should
first verify that the key belongs to the person
named in the user ID. It is useful for others to
know how carefully you verified this.

...

Note that the examples given above for levels 2
and 3 are *only* examples.
In the end, it is up to you to decide just what
"casual" and "extensive"
mean to you when you sign other keys.

If you don't know what the right answer is,
answer "0".

(NIST SP 800-63A)

(0) I will not answer.
You make no particular claim as to how carefully you verified the key.

(1) I have not checked at all.
You believe the key is owned by the person who claims to own it but you could not, or did not verify the key at all. This is useful for a "persona" verification, where you sign the key of a pseudonymous user.

(2) I have done casual checking. (default)
You did casual verification of the key. For example, this could mean that you verified the key fingerprint and checked the user ID on the key against a photo ID.

(3) I have done very careful checking.
You did extensive verification of the key. For example, this could mean that you verified the key fingerprint with the owner of the key in person, and that you checked, by means of a hard to forge document with a photo ID (such as a passport) that the name of the key owner matches the name in the user ID on the key, and finally that you verified (by exchange of email) that the email address on the key belongs to the key owner.

[C]

ACADEMIC USE ONLY!

WEB OF TRUST
(THREAT MODEL)

```
GnuPG needs to construct a user ID to identify your key.

Real name: Admin
Email address: admin@victim.tld
Comment: Official release signing key, supersedes all previous
You selected this USER-ID:
    "Admin (Official release signing key, supersedes all previous) <admin@victim.tld>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit?
```

[C]

WEB OF TRUST
(SIGNING PUBLIC KEYS)

- Import the public key to be signed
- Edit the public key being signed
- Review UID information
- Confirm fingerprint ownership
- Use the **sign** subcommand
- Review confirmation prompt, entering '?' for additional guidance
- Specify a signature level (0-3)
- Enter **save** to commit and exit
- Verify with **`gpg --list-sigs`**
- Export signed public key

```
Your selection? (enter '?' for more information): 3
Are you sure that you want to sign this key with your
key "Matt Borja (First GPG key. For academic use only!)
<me@mattborja.dev>" (E87215B3E9FA14FA)

I have checked this key very carefully.

Really sign? (y/N) Y

gpg> save

$ gpg --list-sigs 01483F262A4B3FF0
pub   rsa4096 2019-09-18 [SC]
      4AF4D83EEDDF43DA3C06CB3101483F262A4B3FF0
uid           [  full  ] Rod Widdowson <rdw@...ware.com>
sig 3        01483F262A4B3FF0 2019-09-18  [self-signature]
sig 3        E87215B3E9FA14FA 2025-07-08  Matt Borja …
sub   rsa4096 2019-09-18 [E]
sig          01483F262A4B3FF0 2019-09-18  [self-signature]
```

WEB OF TRUST
(SIGNING PUBLIC KEYS)

Optional (as requested). Export and return the signed public key directly to owner's UID email:

```
$ THEIR_FINGERPRINT="4AF4D83EEDDF43DA3C06CB3101483F262A4B3FF0"
$ MY_KEY_ID="E87215B3E9FA14FA"

$ gpg --armor --export "$THEIR_FINGERPRINT" > "signedBy-${MY_KEY_ID}.asc"

$ cat ./signedBy-E87215B3E9FA14FA.asc
-----BEGIN PGP PUBLIC KEY BLOCK----
mQINBF2CHx4BEADfUvo4sPc8a8uQjfvoHdBY0qmgwXlcAOyllBKZ5g/wYKZO1Lkp
LZh/dQFBK4AjqRnzs0dq0arK0W5WijOYjQ+s5cd1MMXmzqgXG02eAS4ooK6KsSwv
mo2FydddQKFbwLkGdS/UXcENWNnzGeJhmjmcPSLgRo2hsSh63cFltq2+8fwl1fQ9
FzZwscOOkJGBb7/nqdHdnvL9yrRameYFo2iWF9P52cjfv5NiNklkDBUHeISuX48I
pI+kSOz2b7/aP4vKOKOpOaas0MAdcYT8AcwrCD9OhFFzfuIs/S9+rHGs/+M0vcWG
...
-----END PGP PUBLIC KEY BLOCK-----
```

[S]

SUBKEYS

(DIGITAL SIGNING)

Same keygen steps as before.

- Edit your key
- Use the **addkey** subcommand
- Select a key type for signing
- Toggle only Sign capability
- Select an algorithm for **signing**
- Specify an appropriate expiry
- Confirm
- If needed, provide a passphrase
- Save and review

```
$ gpg --expert --edit-key E87215B3E9FA14FA
gpg (GnuPG) 2.4.7; Copyright (C) 2024 g10 Code GmbH
This is free software: you are free to change and
redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Secret key is available.

sec  ed25519/E87215B3E9FA14FA
     created: 2025-07-06  expires: never       usage: C
     trust: ultimate      validity: ultimate
[ultimate] (1). Matt Borja (... For academic use only!)

gpg> addkey
[see instructions at left]

sec  ed25519/E87215B3E9FA14FA
     created: 2025-07-06  expires: never       usage: C
     trust: ultimate      validity: ultimate
ssb  ed25519/B487BEEC47AF1237
     created: 2025-07-09  expires: 2026-07-09  usage: S
[ultimate] (1). Matt Borja (.. For academic use only!)
```

[S]

SUBKEYS
(DIGITAL SIGNING)

- Sign a message (`--clearsign`)
- Sign a file (`--armor --detach-sign`)

```
$ gpg --clearsign -u E87215B3E9FA14FA
Hello, world!^D

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

Hello, world!
-----BEGIN PGP SIGNATURE-----

iHUEARYKAB0WIQSRmwkjFAeFgtJAmM+0
h77sR68SNwUCaG4I4gAKCRC0h77sR68S
N1byAP9dwhknuMKhKDWKLQ7xHExy9pi+
9Ji+IQHMYcAmQfzUzwD/dwDs07Ey0MTZ
3IxmGkWJklsrCDXOUz4Z7i/X/f9ggAg=
=lGAg
-----END PGP SIGNATURE-----


$ gpg --armor --detach-sign -u E87215B3E9FA14FA file.ext

$ ls file.ext*
file.ext      file.ext.sig
```

Repeat for **[E]ncryption** type subkeys.

NEXT STEPS

- **B2B safeguard.** Edit your key to add an **Encryption** subkey for exchanging sensitive information with trusted parties (e.g. sharing API keys or login credentials with third-party vendors).
- **Supply chain safeguard.** Verify every download accompanied with a detached signature.
- **Release.** Incorporate digital signing into your release management (e.g., binary packages, software provenance, et. al.).
- **Backup and restore.** Learn how to use `paperkey` for creating and restoring your private key using paper backups.
- **Personal safeguard.** Practice the use of your **Sign** and **Encrypt** subkeys (e.g., email).
- **Secure logins.** Edit your key to add an **Authentication** subkey to reconfigure SSH authentication with (e.g., `git,` SSH tunnel).

NEXT STEPS

- ⭐ **Highly recommended.** Upgrade to a YubiKey with OpenPGP support as **a preferred smart card** (hardware) alternative for key management, encryption, signing and authentication operations.
  Doubles as a phishing-resistant multi-factor authentication (MFA) device!
- ⭐ **Build a secure environment.** Don't have a YubiKey?
  Learn more about *Creating a Secure Environment for GPG in Alpine Linux*.
- **Build your web of trust.** Participate in key signing [parties] with people you know!
- **Advocate.** Write your own **key signing policy**.
- **Support identity assurance.** Find and contribute reliable identity evidence for key verifications at **https://SIG3.dev**

Matt Borja

me@mattborja.dev

TOPICS:

- Identity Management
- Information Security
- Software Engineering